FIG. 1

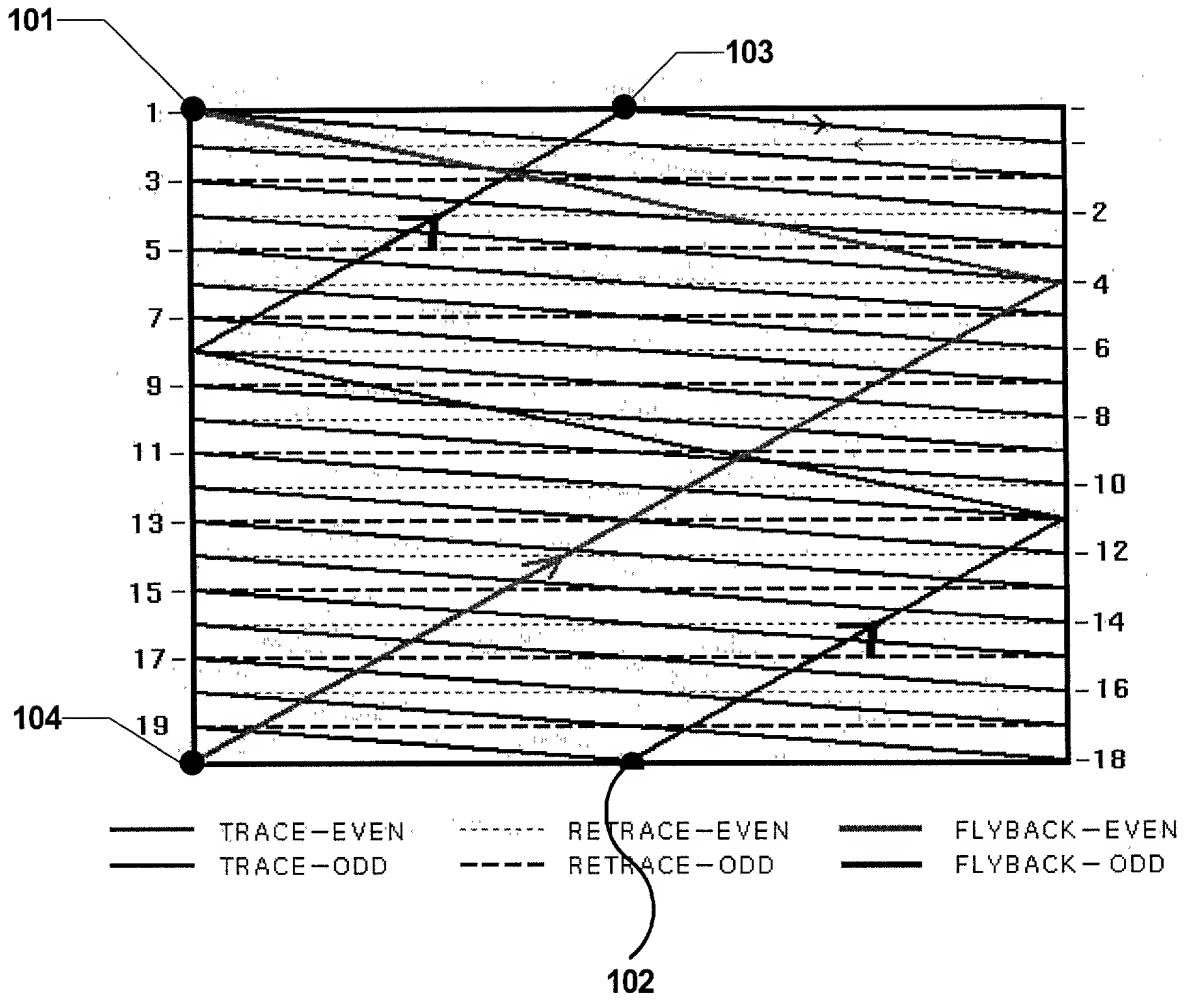

101
103
104
102
1
3
5
7
9
11
13
15
17
19
2
4
6
8
10
12
14
16
18
TRACE—EVEN
TRACE—ODD
RETRACE—EVEN
RETRACE—ODD
FLYBACK—EVEN
FLYBACK—ODD

FIG. 2

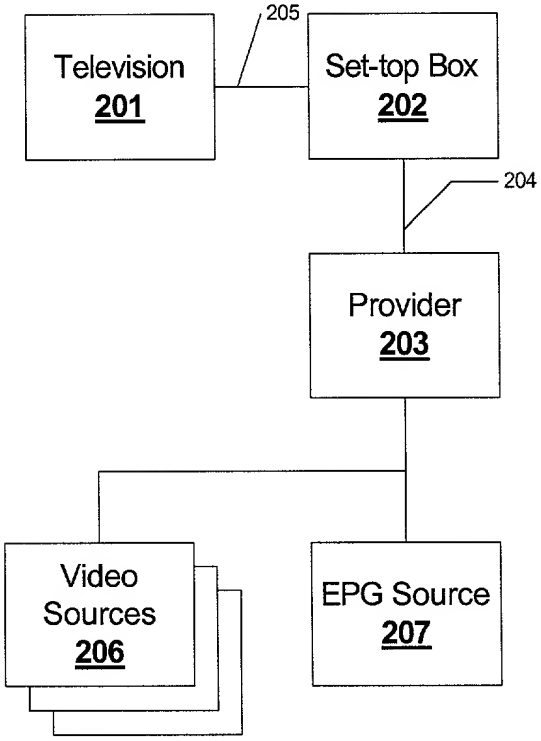

Television
201
205
Set-top Box
202
204
Provider
203
Video
Sources
206
EPG Source
207

FIG. 3A

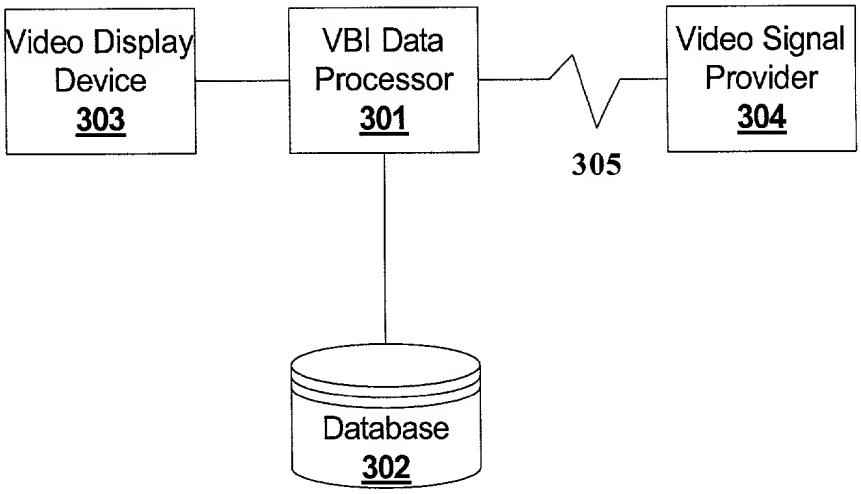

Video Display Device
303
VBI Data Processor
301
Video Signal Provider
304
305
Database
302

FIG. 3B

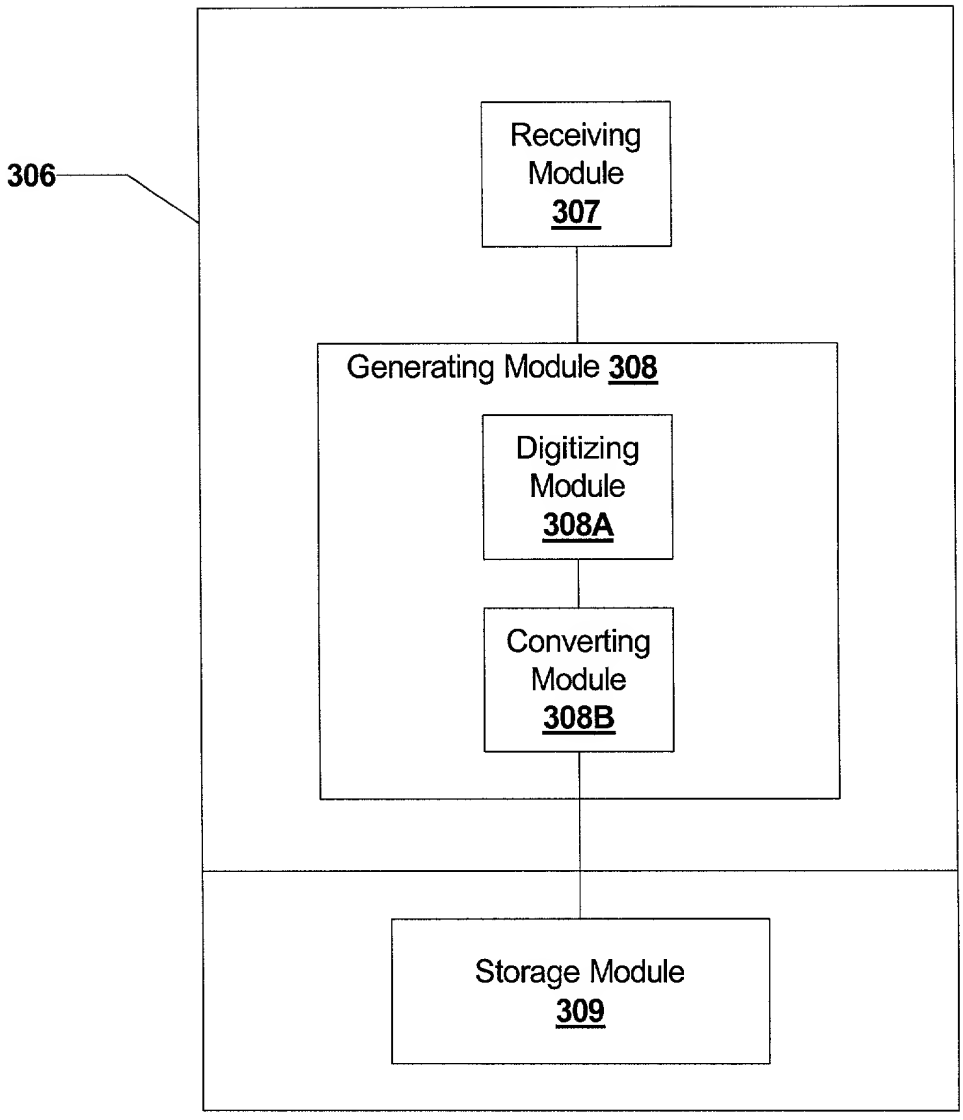

306
Receiving Module 307
Generating Module 308
Digitizing Module 308A
Converting Module 308B
Storage Module 309

FIG. 4

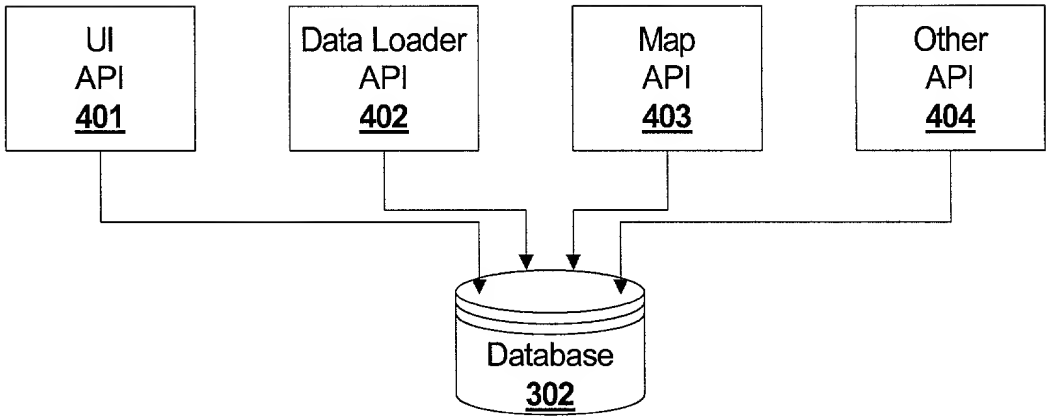

UI
API
401
Data Loader
API
402
Map
API
403
Other
API
404
Database
302

FIG. 5

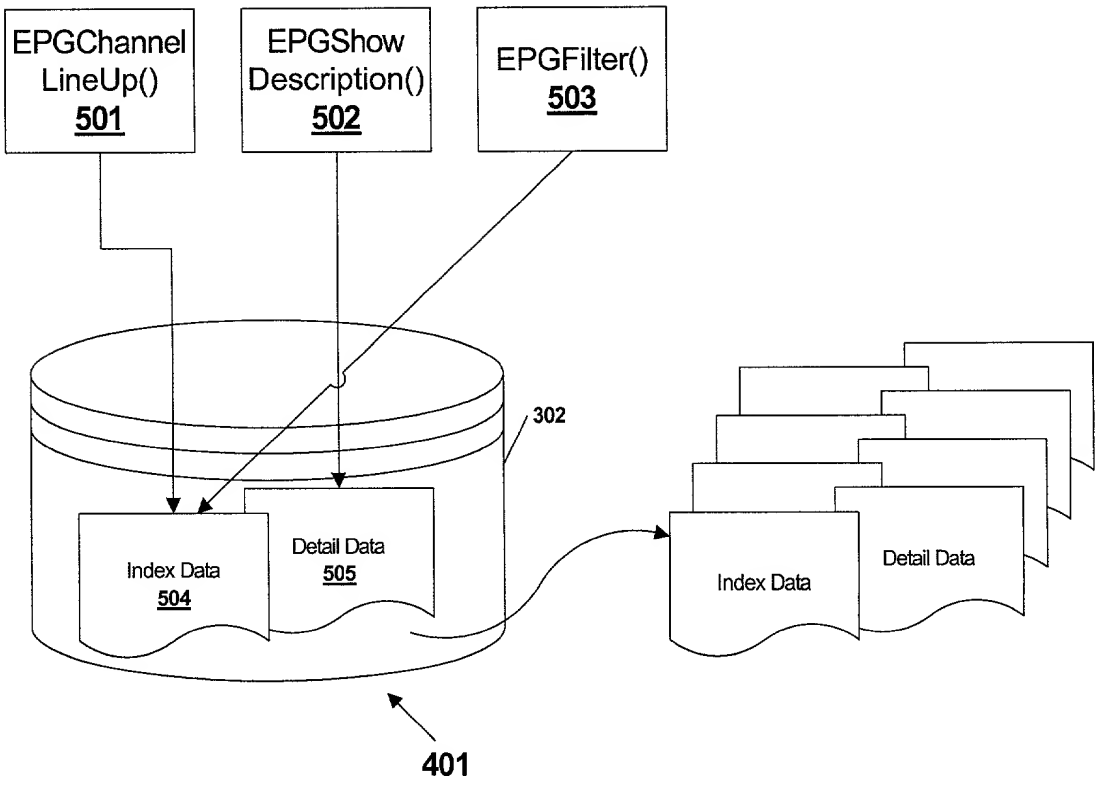

EPGChannel LineUp() 501
EPGShow Description() 502
EPGFilter() 503
302
Index Data 504
Detail Data 505
Index Data
Detail Data
401

FIG. 6A

API for UI

1. EPGChannelLineup

Description Call this function to get a list of brief show information for a given range of channel numbers and time frame. The returned show array is sorted in the order of channel number and show beginning time.

Prototype Boolean EPGChannelLineup(const line_up & sline, show_info * const sinfo, int * const sorder, int &length, query_error & err);

Parameters

- sline: IN PARAMETER. This structure specifies time grids and the range of channel numbers. The data structure is defined as:

```
struct line_up
{
    time_t time_grid_start;
    time_t time_grid_end;
    int channel_id_start;
    int channel_id_end;
}
```

- sinfo: OUT PARAMETER. sinfo is an array of type show_info.

```
struct show_info
{
    int channel_number;
    char call_letter[9];
    time_t begin_time;
    short duration;
    unsigned char cat_index;
    unsigned char sub_cat_index;
    char short_title[22];
    long reference_number;
}
```

- sorder: OUT PARAMETER. It stores the order of shows sorted in channel number and time. For example, sinfo[sorder[0]] is the first show, sinfo[sorder[1]] the second, etc.
- length: IN OUT PARAMETER. As an in parameter, length is the dimension of sinfo and sorder, as an out parameter, it is the actual number of shows stored in sinfo and sorder, provided the number is less than the input length. Otherwise, FALSE is returned and err is populated with messages.
- Err: OUT PARAMETER. This parameter holds error information if error occurs. Non-zero err_code means something wrong in the call. Query_error is defined as

```
struct query_error
{
    int err_code;
    char err_msg[100];
}
```

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading

2. EPGShowDescription

Description This function returns a detailed show data. In order to get detailed show info, caller needs to pass the beginning time (or date) and reference number of the show. Reference number of a show is obtained by calling EPGChannelLineup function.

Prototype Boolean EPGShowDescription(show_brief_info &sbrief, show_description & sdesc, query_error & err);

Parameters

- sbrief: IN PARAMETER. The struct show_brief_info is defined as

```
{
    time_t date;
    long reference_number;
```

Patent Application Publication Sep. 27, 2007 Sheet 9 of 9 US 2007/0226759 A1

FIG. 6B

}

- sdesc: OUT PARAMETER. Detailed description of a show, including category, description, etc. The data structure is defined as

```
struct show_description
{
    char rest_of_title[100];
    char short_description[64];
    char description[801];
    char category[13];
    char subcategory[13];
    short year_produced;
    float stars;
    bool re_run;
    bool live;
    bool closed_caption;
    bool stereo;
    char tv_rating[13];
    char mpaa_rating[5];
}
```

- err: OUT PARAMETER.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

3. EPGFilter

Description To get a list of shows having specified category-index and subcategory-index. Category-index and subcategory-index are numbers between 0 to 15.

Prototype Boolean EPGFilter(show_cat_info &scat, show_info * const sinfo, int * const sorder, int &length, query_error & err);

Parameters

- scat: IN PARAMETER. The data struct show_cat_info is defined as

```
struct show_cat_info
{
    short cat_index;
    short sub_cat_index;
    time_t begin_time;
    unsigned char updown_flag;
}
```

  Above data structure specifies category index, subcategory index, beginning time of a search, and forward or backward search flag. updown_flag=1 means forward search, 0 means backward search. The search is limited among the shows of the same day as the specified beginning time.
- sinfo: OUT PARAMETER. sinfo is an array of type show_info.
- sorder: OUT PARAMETER. It stores the order of shows sorted in channel number and time. For example, sinfo[sorder[0]] is the first show, sinfo[sorder[1]] the second, etc.
- length: IN OUT PARAMETER. As an in parameter, length is the dimension of sinfo and sorder, as an out parameter, it is the actual number of shows stored in sinfo and sorder, provided the number is less than the input length. Otherwise, FALSE is returned and err is populated with messages.
- Err: OUT PARAMETER. This parameter holds error information if error occurs. Non-zero err_code means something wrong in the call.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

FIG. 7

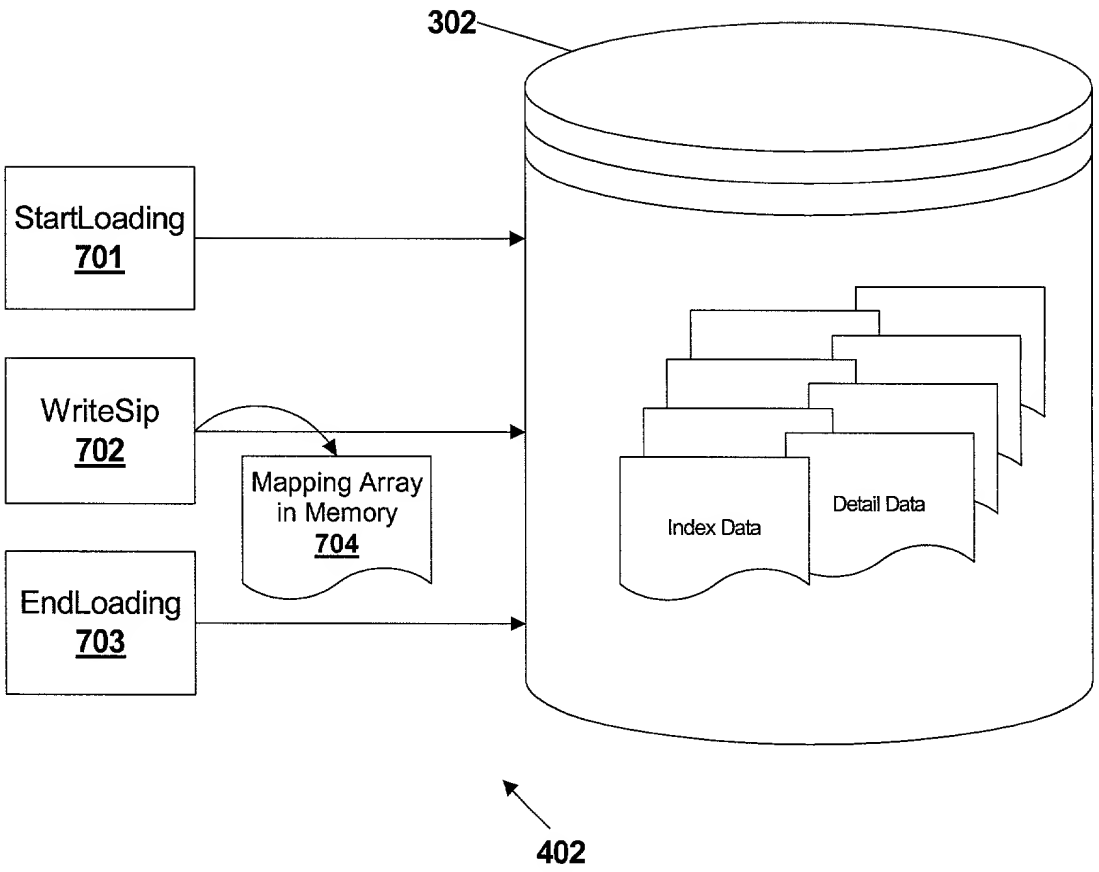

302
StartLoading
701
WriteSip
702
Mapping Array
in Memory
704
EndLoading
703
Index Data
Detail Data
402

FIG. 8A

API for Data Loader

1. TsipCallback::StartLoading

Description StartLoading is responsible for preparing database tables for the data loading. Therefore, this function must be called before data loading. After data loading is done, EndLoading function shall be called.

Prototype Boolean StartLoading()

Parameters None

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

2. TsipCallback::StartSip

Description StartSip marks the beginning of the data loading of a show information packet. A show information packet contains 4-hour show data for one channel. EndSip indicates the end of the data loading of one packet.

Prototype Boolean StartSip()

Parameters None

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

3. TsipCallback::WriteSip

Description WriteSip inserts one or couple of show data into database tables.

Prototype Boolean WriteSip(const SipHeader &shdr, const Showinfo *psi, int length)

Parameters

- shdr: IN PARAMETER. SipHeader contains common data shored by show data stored in the array Showinfo.

```
struct
TUNECHAN {
    unsigned char type;          // b0-2: 0:OTA 1:cable 2:satellite 3-7:reserved
                                 // b3:digital b4:dual A/B cable trunk b5-7:rsvd
    unsigned short minor;        // up to 10 bits of digital minor channel
    unsigned short major;        // up to 10 bits of digital major channel, or analog channel
}
struct SipHeader
{
    short channel_id;            // unique channel ID
    TUNECHAN tune_channel;  // see struct definition above
    char channel_name[9];        // (null terminated string)
    unsigned char day_of_week;  // 0,1,2,6 for Sun, ..., Saturday, respectively
    char date[9];                // YYYYMMDD null terminated
                                 // on 4 hour intervals (0, 4, 8, 12, 16, 20)
}
```

- psi: IN PARAMETER. Showinfo is defined as

```
struct Showinfo
{
    char short_title[22];        // short title, as displayed on grid titles
    char rest_of_title[100];     //concatenated with short title, the complete show title
    char short_description[64]; // short description. Enough descriptive text to fit 3x21
                                 // display. (includes ratings for movies, teams for sports,
                                 // subjects for talk shows, etc. char null terminated.
    char long_description[801]; // with short description, complete description
                                 //  null terminated
    char begin_time[15];
```

FIG. 8B

```
                // Show starting time with format of YYYYMMDDHHMI
                // null terminated.  MM takes a value of 01, ..., 12;
                // DD takes a value of 01, ..., 31; HH takes a value of 00, ..., 23;
                // MI takes a value from 00 to 59.
        char end_time[15];  // as above, used for performance purpose
        int duration; // in minutes
        char category[13];  // category name, null terminated
        unsigned char cat_index;
        char subcategory[13];       // sub-category name, null terminated
        unsigned char sub_cat_index;
        short year_produced;        // four digits, e.g., 1998.  For search purpose
        float stars;                // 0, 0.5, 1, 1.5, 2, ..., 4.5, 5 for search purpose
        bool re_run;
        bool live;
        bool closed_caption;
        bool stereo;
        char TV_rating[13];  // "TVMA-FVDLVS", "TVY-LV", ..., null terminated
        char mpaa_rating[5]; // "NC17", "X", ..., null terminated
    };
    psi is the pointer of the array Showinfo psi[length].
```

- length: IN PARAMETER. length tells how many show records in the array psi.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

4. TsipCallback::EndSip

Description EndSip marks the end of the data loading of a show information packet.
Prototype Void EndSip()
Parameters None
Return Value Void

5. TsipCallback::EndLoading

Description After data loading is successfully completed, Data Loader calls EndLoading to invoke a sequence of actions on the database tables, such as clean up temporary tables and etc. If, for some reason, this API is not called, then show data for UI will not be updated.
Prototype Void EndLoading()
Parameters None
Return Value Void

FIG. 9

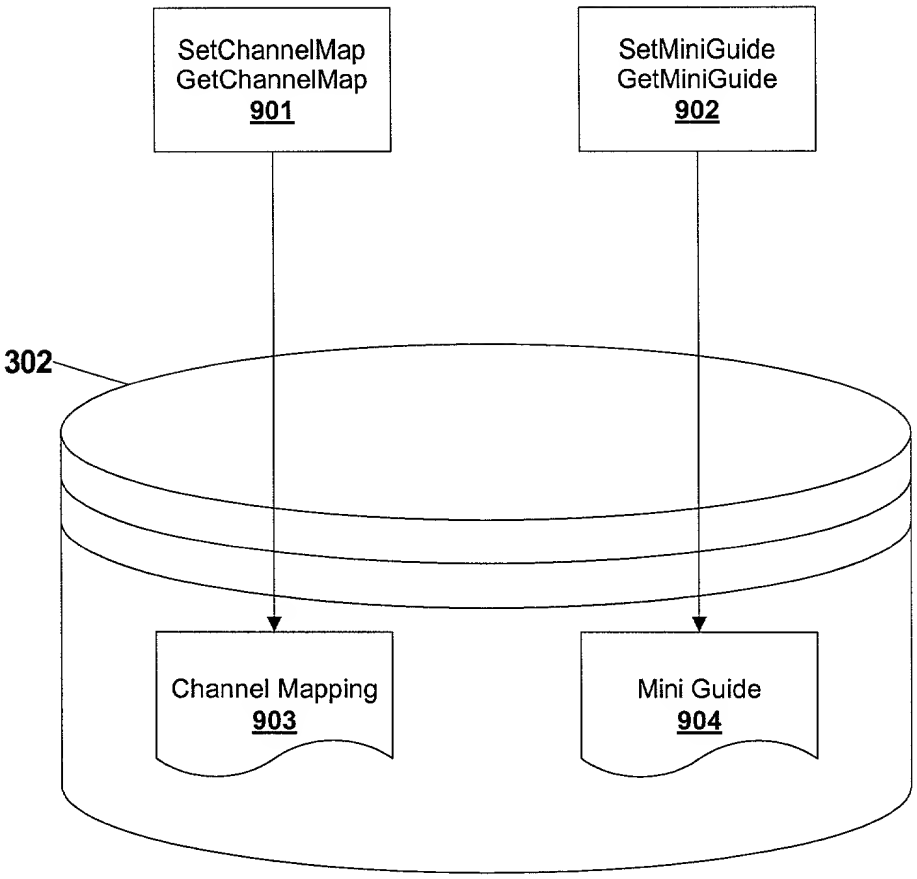

SetChannelMap
GetChannelMap
901
SetMiniGuide
GetMiniGuide
902
302
Channel Mapping
903
Mini Guide
904

FIG. 10A

API for Channel Mapping

1. **MapGet**

Description This function returns an array of Channel Mapping data to the caller. Channel Mapping describes the mapping among Channel, Call letter, Category, and Channel Id.

Prototype Boolean MapGet(channel_map * const minfo, int * const sorder, int &length, query_error & err)

Parameters

- minfo: OUT PARAMETER. The pointer minfo points to an array of channel_map. The struct channel_map is defined as

```
struct channel_map
{
    char call_letter[9];
    unsigned short channel id;
    unsigned short channel;
    unsigned short cat;
};
```

- sorder: OUT PARAMETER. It stores the order of Channel Mappings sorted in channel number.
- length: IN OUT PARAMETER. As an in parameter, length is the dimension of minfo and sorder, as an out parameter, it is the actual number of shows stored in minfo and sorder, provided the number is less than the input length. Otherwise, FALSE is returned and err is populated with messages.
- Err: OUT PARAMETER. This parameter holds error information if error occurs. Non-zero err_code means something wrong in the call.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

2. **MapSet**

Description Caller can set Channel Information by calling this function. Inside of MapSet, it checks each element of the array minfo against the records in the database. If the call letter exists in the database, then update the row by the new data from minfo; if call letter doesn't exist in the database, then insert a new row into the database.

Prototype Boolean MapSet(channel_map * const minfo, int &length, query_error & err)

Parameters

- minfo: IN PARAMETER. The parameter minfo is an array of channel_map. The struct channel_map is defined as

```
struct channel_map
{
    char call_letter[9];
    unsigned short channel id;
    unsigned short channel;
    unsigned short cat;
};
```

- length: IN PARAMETER. It is the actual number of Channel Mappings stored in minfo.
- Err: OUT PARAMETER. Non-zero err_code means something wrong in the call.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

3. **MiniGuideGet**

Description To get Mini Guide data. Mini Guide shows the mapping between Category and Channel. At present, there are about 12 categories. Each Category owns a segment of Channels.

Prototype Boolean MiniGuideGet(mini_guide * const minfo, int * const sorder, int &length, query_error & err)

FIG. 10B

Parameters

- minfo: OUT PARAMETER. minfo is an array of mini_guide. The data struct *mini_guide* is defined as

```
struct
{
    char name[15];              // Category
    unsigned short channel;     // Channel Number
    unsigned short offline;     // Channel number starts at
    unsigned short start;       // Channel segment starts at
    unsigned short end;         // Channel segment ends at
    unsigend short dup_start;   // Local Channel starts at
}
```

- sorder: OUT PARAMETER. The order of Mini Guides by channel number.
- length: IN OUT PARAMETER. As an in parameter, length tells the dimension of minfo and sorder, as an out parameter, it is the actual number of Mini Guides stored in minfo and sorder, provided the number is less than the input length. Otherwise, FALSE is returned.
- Err: OUT PARAMETER. Non-zero err_code means something wrong in the call.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

4. **MiniGuideSet**

Description Caller can set Mini Guides by calling this function. Inside of MiniGuideSet, each element of the array minfo is checked against the records in the database. If a category exists in the database, then update the row by the new data from minfo; if a category doesn't exist in the database, then insert a new row into the database.

Prototype Boolean MiniGuideSet(mini_guide * const minfo, int &length, query_error & err)

Parameters

- minfo: IN PARAMETER. The pointer minfo points to an array of mini_guide.
- length: IN PARAMETER. It is the actual number of Mini Guides stored in minfo.
- Err: OUT PARAMETER. Non-zero err_code means something wrong in the call.

Return Value TRUE: Success
FALSE: Fails to prepare tables for data loading.

FIG. 11
(Partial Software)

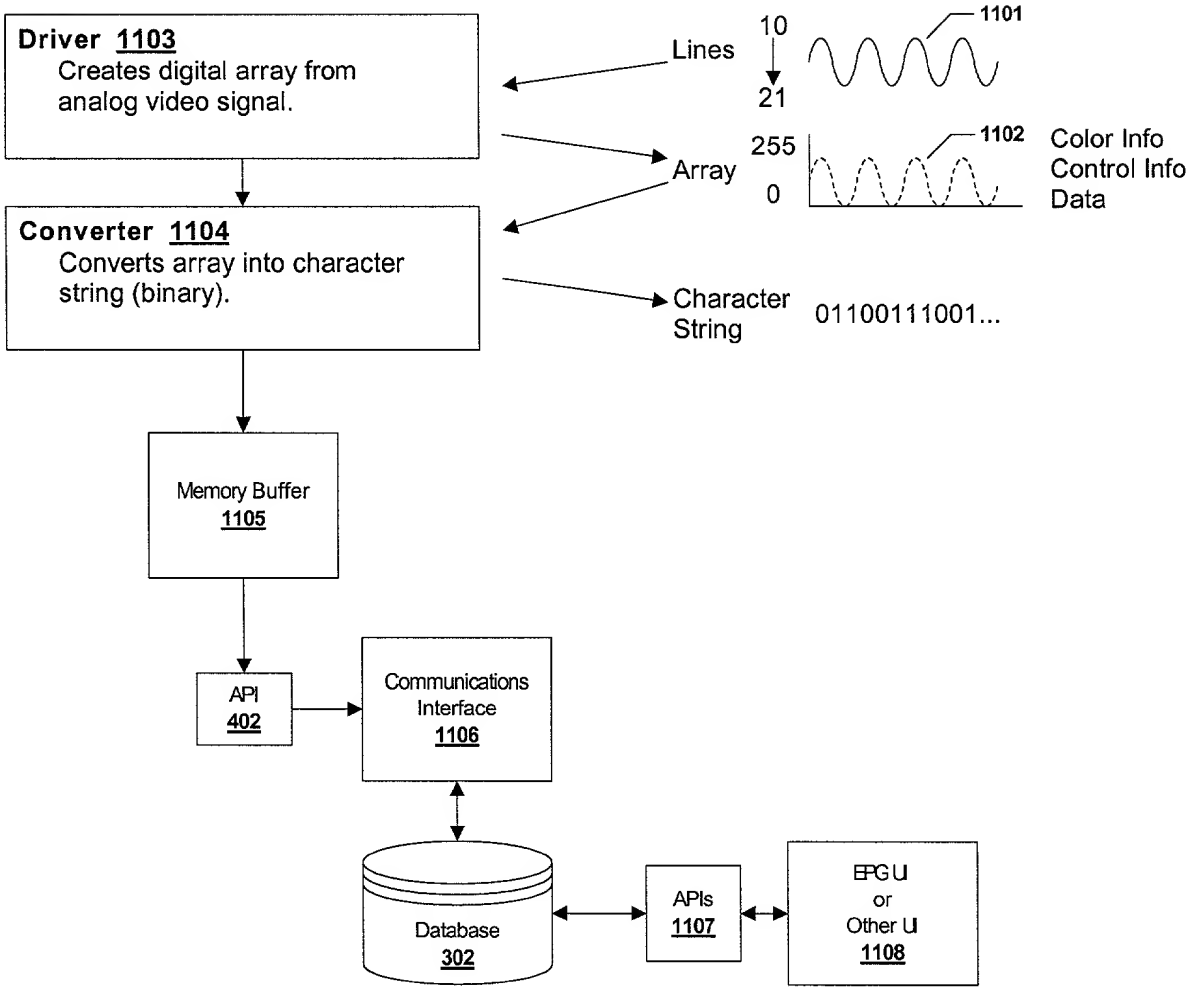


FIG. 12

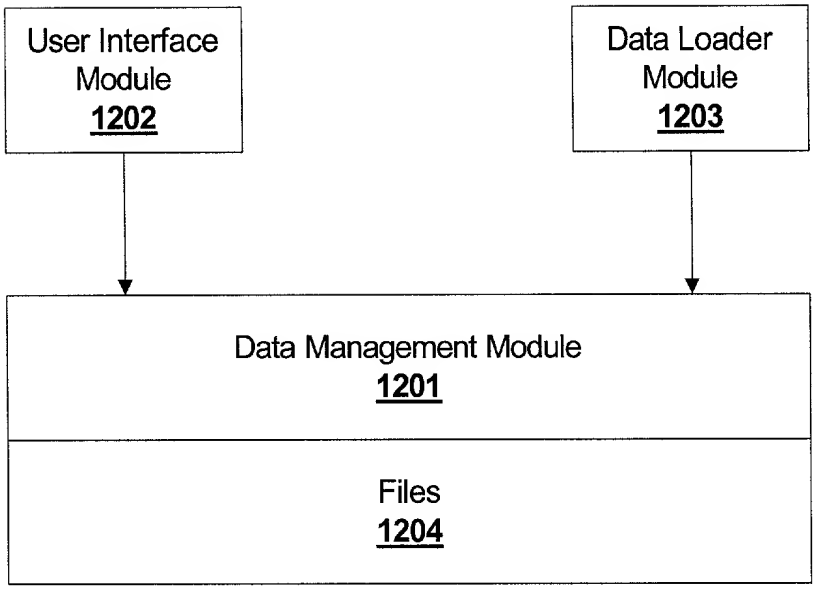

User Interface Module
1202
Data Loader Module
1203
Data Management Module
1201
Files
1204

FIG. 13

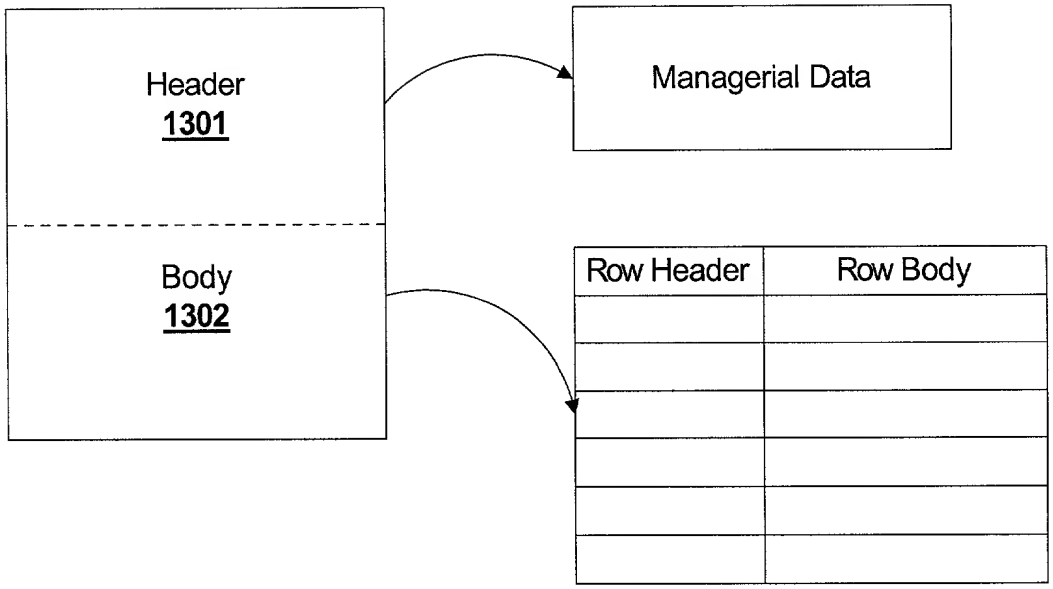


FIG. 14

```
struct adr_hdr;
{
        char_name[30];
        char f_name[30];
        char c_date[15];
        long t_row;
        long a_row;
        long na_row;
        short hdr_len;
        short row_start;
        short row_len;;
        short row_hdr_len;
        short row_bdy_len;
        short row_hdr_start;
};
```

where

- Name: Table name
- C_date: When the table is created, in the form "YYYYMMDDHHMISS"
- F_name: the file where the data is stored
- T_row: Total number of initialized rows
- Na_row: Next available row id for insertion
- Hdr_len: length of the header
- Row_start: where the data (row) start (=hdr_len);
- Row_len: the length of a row
- Row_hdr_len: length of the row header
- Row_bdy_len: the length of the row-body
- Row_hdr_start: Where the row-header starts

```
Struct row_hdr
{
        long row_id;
        long next_available_row;
        char usage;
}
```

where

- Row_id: row id of the row. Increased by one each time
- Next_avaialble_row: row id of the next available row for insertion.
- usage: 'N' means available; 'Y' means used.

FIG. 15

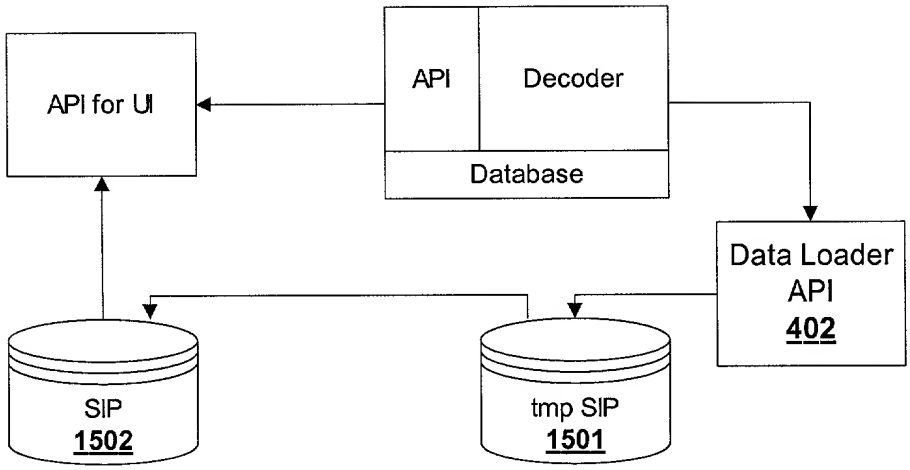
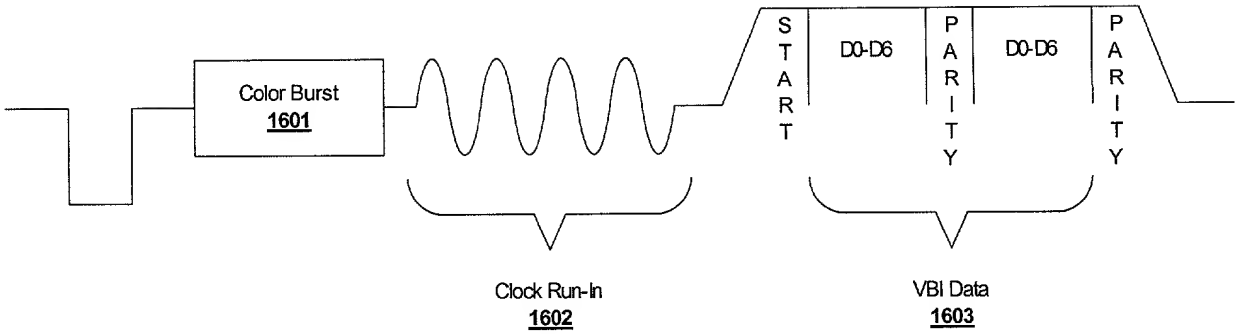

API for UI
API
Decoder
Database
Data Loader
API
402
SIP
1502
tmp SIP
1501

FIG. 16


Color Burst
1601
START
D0-D6
PARITY
D0-D6
PARITY
Clock Run-In
1602
VBI Data
1603